# CALIDAD DEL VINO TINTO

## 1. CONTEXTO

Se trabaja con un dataset de la variante tinta del "Vinho verde" del norte de Portugal (para más detalles, consultar http://www.vinhoverde.pt/en/) porque, ¿a quién no le gusta el vino?. El objetivo es **modelar la calidad del vino** basándose en pruebas físico-químicas (ver [Cortez et al., 2009], http://www3.dsi.uminho.pt/pcortez/wine/). Debido a cuestiones de privacidad y logística, solo están disponibles variables fisicoquímicas (entradas) y sensoriales (salidas) (por ejemplo, no hay datos sobre tipos de uva, marca de vino, precio de venta del vino, etc.). Link al dataset: https://archive.ics.uci.edu/dataset/186/wine+quality.

El objetivo principal es **recopilar las características del vino que contribuyen a una mejor calidad del mismo** (identificación de alguna variable que no aporta mucho por ejemplo).

El tamaño del conjunto de datos es de 1599 muestras, que tienen doce campos, donde once campos son para la entrada y uno para salida.

Acidez fija, Acidez volátil, Ácido cítrico, Azúcar residual, Cloruros, Dióxido de azufre libre, Dióxido de azufre total, Densidad, pH, Sulfatos y Alcohol representan los campos de entrada.

El campo de salida corresponde a la calidad del vino, **que es un puntaje entre 0 y 10**.

### Columnas en el Dataset

Variables de entrada (basadas en pruebas físico-químicas):

- fixed acidity: Acidez fija
- volatile acidity: Acidez volátil
- citric acid: Ácido cítrico
- residual sugar: Azúcar residual
- chlorides: Cloruros
- free sulfur dioxide: Dióxido de azufre libre
- total sulfur dioxide: Dióxido de azufre total
- density: Densidad
- pH: pH
- sulphates: Sulfatos
- alcohol: Alcohol

Variable de salida (basada en datos sensoriales):

- quality: Calidad (puntaje entre 0 y 10)

## 2. PREPROCESAMIENTO DE DATOS

Módulos y librerías útiles:

```
# Importación, visualización, manipulación de datos
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Transformación de datos
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Métricas y desempeño
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

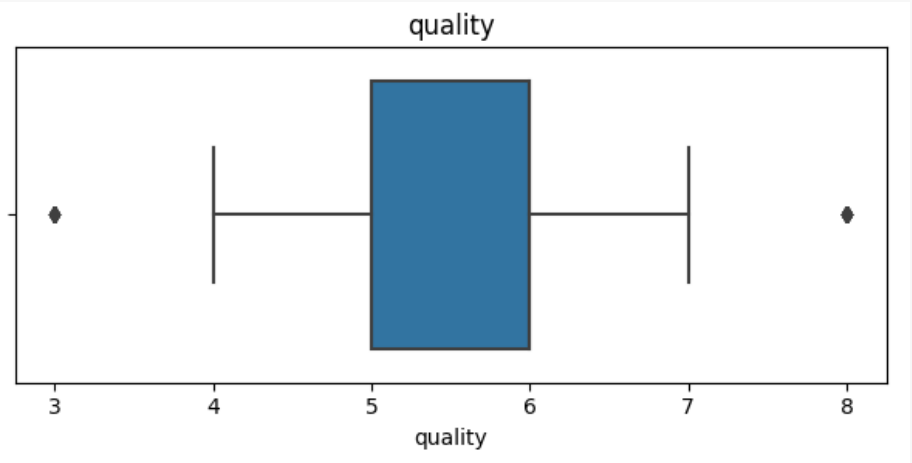
Mediante *dataset.info()* se observa que **todas las variables son numéricas** (las de entrada flotantes y la de salida 'quality' entera).

Afortunadamente **no hay datos nulos** en el dataset, como se puede ver con *dataset.isna().sum().sort_values(ascending = False)*.

Sin embargo, vemos con *dataset.duplicated().sum()* que **hay 240 filas con la misma información**, por lo que se eliminan las repetidas mediante *dataset.drop_duplicates(inplace=True)*.

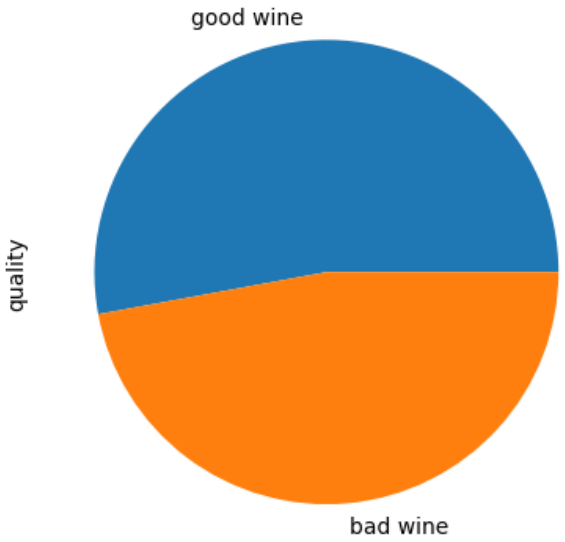
Al utilizar *dataset.describe()* vemos que el puntaje mínimo de 'quality' (calidad) es 3 y el puntaje máximo es 8, siendo la media de aproximadamente 5,62.

Esto nos da la idea de que podemos **clasificar en 'good wine' (vino bueno) a aquellos de puntaje mayor o igual a 6 y 'bad wine' (vino malo) a aquellos de puntaje menor o igual a 5.**

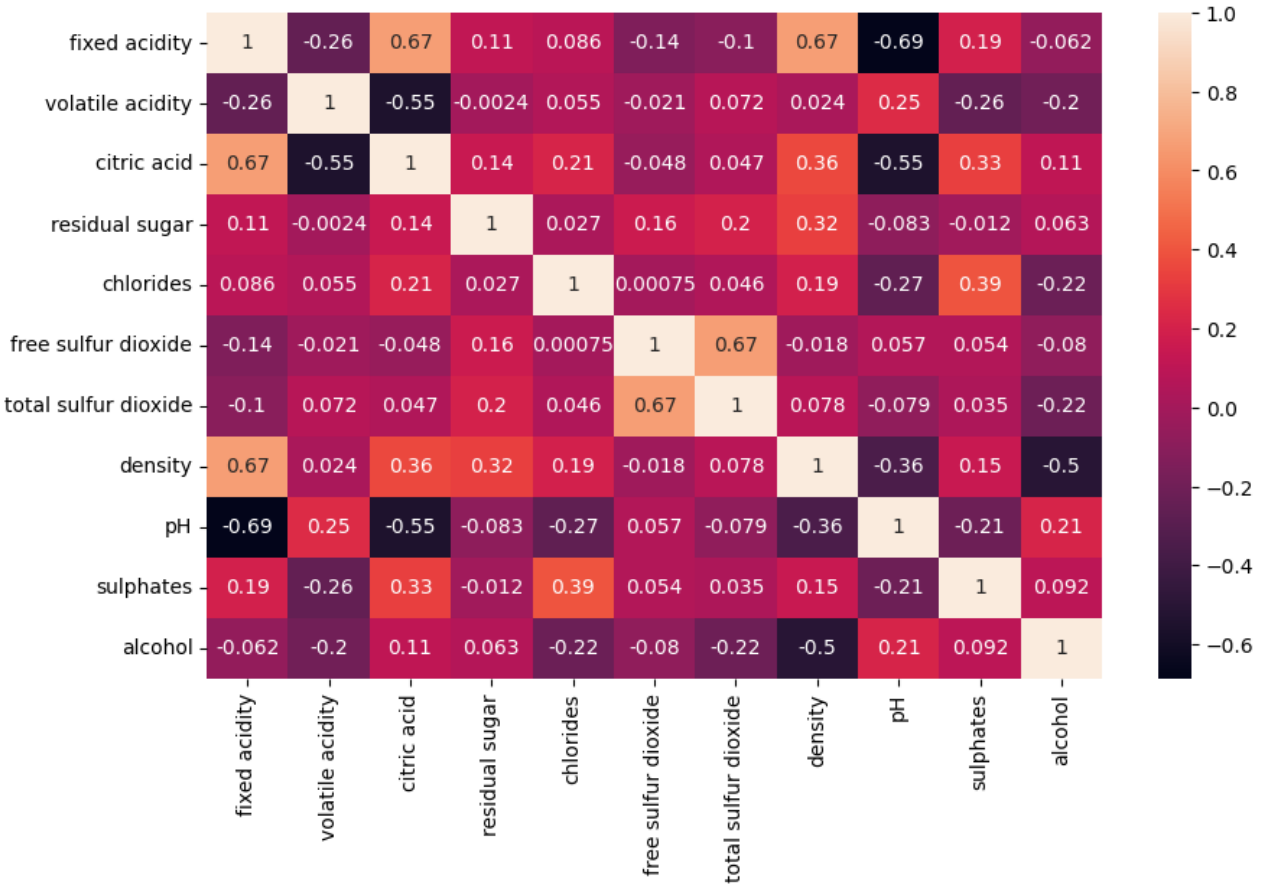



Hago esto último mediante *dataset['quality'] = dataset['quality'].apply(lambda x: 'good wine' if x > 5.5 else 'bad wine')*.

Para saber cuál es la tasa de vinos buenos y vinos malos uso *dataset.quality.value_counts(normalize = True) * 100*, resultando en aproximadamente un 53% de vinos buenos y un 47% de vinos malos, por lo que esto **está balanceado**.




A continuación, se genera la siguiente matriz de correlación:

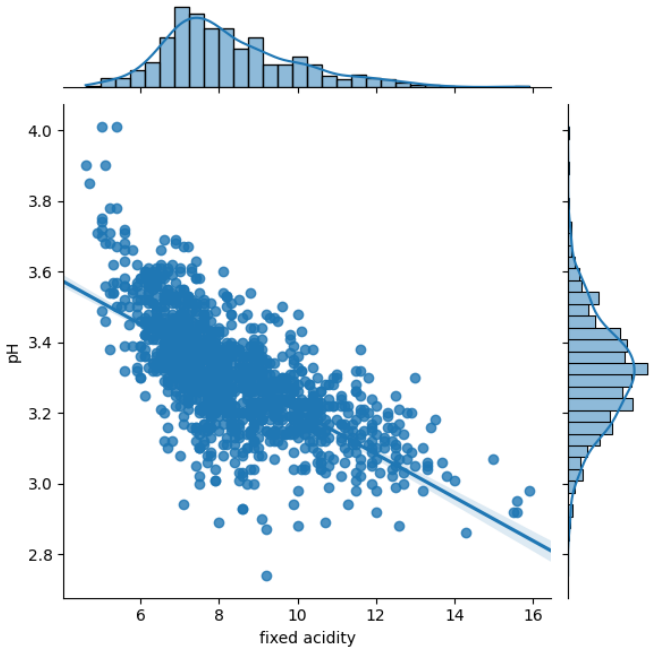


Vemos que las variables más correlacionadas son:

- 'fixed acidity' y 'pH' (-0.69)
- 'fixed acidity' y 'citric acid' (0.67)
- 'fixed acidity' y 'density' (0.67)
- 'free sulfur dioxide' y 'total sulfur dioxide' (0.67)
- 'volatile acidity' y 'citric acid' (-0.55)
- 'citric acid' y 'pH' (-0.55)
- 'density' y 'alcohol' (-0.5)

Concluimos que en primer lugar **podríamos eliminar las variables 'fixed acidity' y 'citric acid'** por estar altamente correlacionadas con las demás.

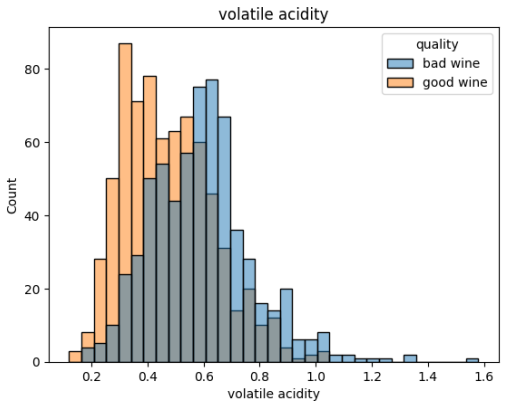
En el siguiente gráfico se ve la correlación entre ‘fixed acidity’ y ‘pH’:

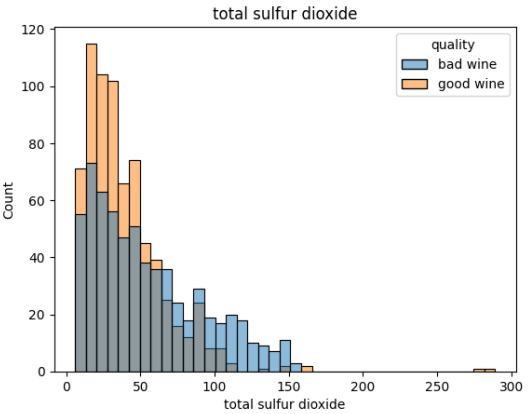


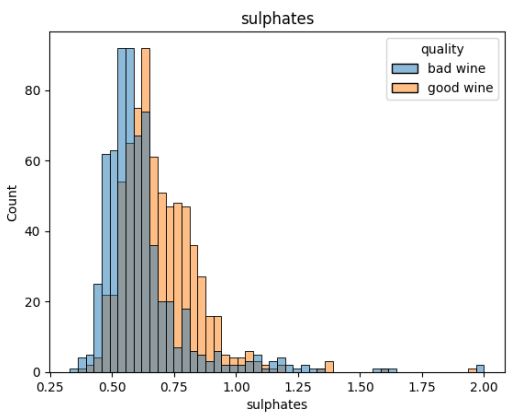

En segundo lugar **podríamos por ejemplo eliminar las variables 'free sulfur dioxide'** (que está correlacionada con 'total sulfur dioxide') **y 'density'** (que está correlacionada con 'alcohol').

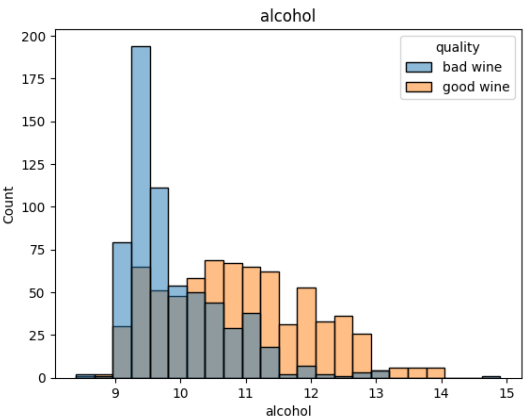
Sin embargo, debido a que las correlaciones no son tan altas (un máximo de 69%) **por ahora trabajaremos con el dataset completo**.

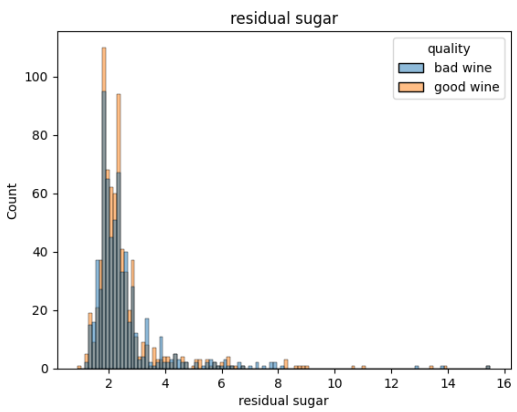
A continuación se grafican algunas distribuciones respecto a la salida:

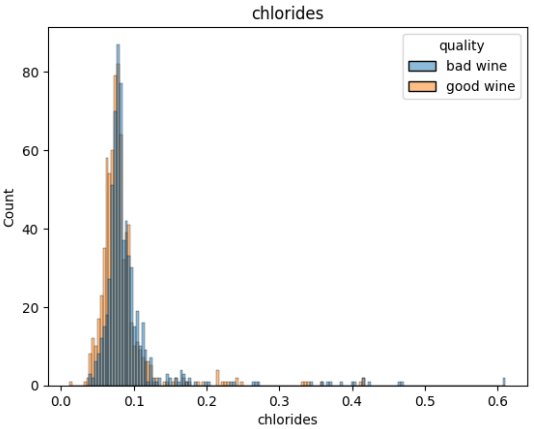

volatile acidity
quality
bad wine
good wine
Count
volatile acidity

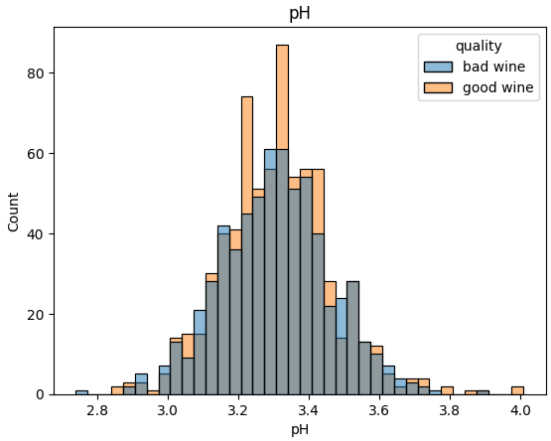

total sulfur dioxide
quality
bad wine
good wine
Count
total sulfur dioxide


sulphates
quality
bad wine
good wine
Count
sulphates


alcohol
quality
bad wine
good wine
Count
alcohol


residual sugar
quality
bad wine
good wine
Count
residual sugar


chlorides
quality
bad wine
good wine
Count
chlorides


pH
quality
bad wine
good wine
Count
pH

Pareciera que:

- A menor cantidad de 'volatile acidity' y 'total sulfur dioxide', mejor la calidad del vino.
- A mayor cantidad de 'sulphates' y de 'alcohol', mejor la calidad del vino.

Por lo que **estas 4 variables deberían ser de importancia**.

En cambio las variables 'residual sugar', 'chlorides' y 'pH' pareciera que no aportan demasiada información acerca de la calidad del vino.

Todo esto es confirmado mediante la siguiente tabla de información mutua de cada variable de entrada con la variable de salida ‘quality’:

| Index | Variables | Información mutua |
|---|---|---|
| **10** | alcohol | 0.150781 |
| **1** | volatile acidity | 0.062584 |
| **9** | sulphates | 0.055823 |
| **6** | total sulfur dioxide | 0.050908 |
| **2** | citric acid | 0.034267 |
| **4** | chlorides | 0.024135 |
| **7** | density | 0.018335 |
| **5** | free sulfur dioxide | 0.017006 |
| **0** | fixed acidity | 0.010272 |
| **8** | pH | 0.005843 |
| **3** | residual sugar | 0.0 |

Observemos que **las 4 primeras variables son las que previamente consideramos más relevantes** para la variable de salida y las 7 siguientes son las que descartábamos por correlación o por ser poco relevantes para la variable de salida.

Nota de color: La más relevante de todas es la variable ‘alcohol’, ¿por qué será? 😊

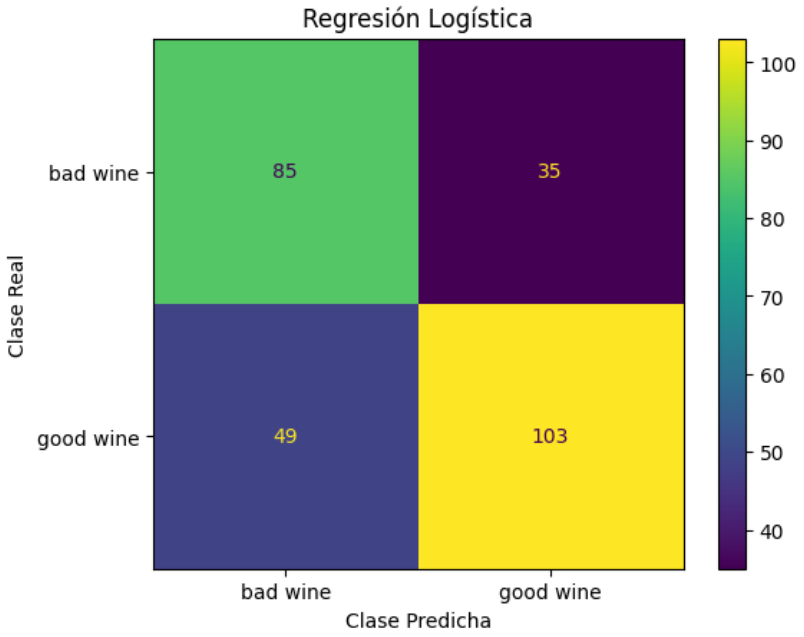
Previo a esta tabla, se copió el dataset mediante *dataset_ml = dataset.copy()*, con *y = dataset_ml.pop('quality')* se define la columna de salida con *le = LabelEncoder()* se codifica con 0 y 1 a las categorías ‘bad wine’ y ‘good wine’ respectivamente y finalmente con *y = le.fit_transform(y)* se recupera luego la etiqueta original.

Por último, mediante *StandardScaler* se realiza el **escalado de datos numéricos** y se genera *dataset_stan* con el mismo.

## 3. PROBLEMA DE CLASIFICACIÓN

En primer lugar se separa al dataset en un **80% para entrenamiento** y un **20% para test**.

A continuación, se utiliza un modelo de Regresión Logística obteniendo un F1-score de 69,22%.




Vemos que 49 vinos fueron predichos como malos cuando en realidad eran buenos, y 35 vinos fueron predichos como buenos cuando en realidad eran malos.

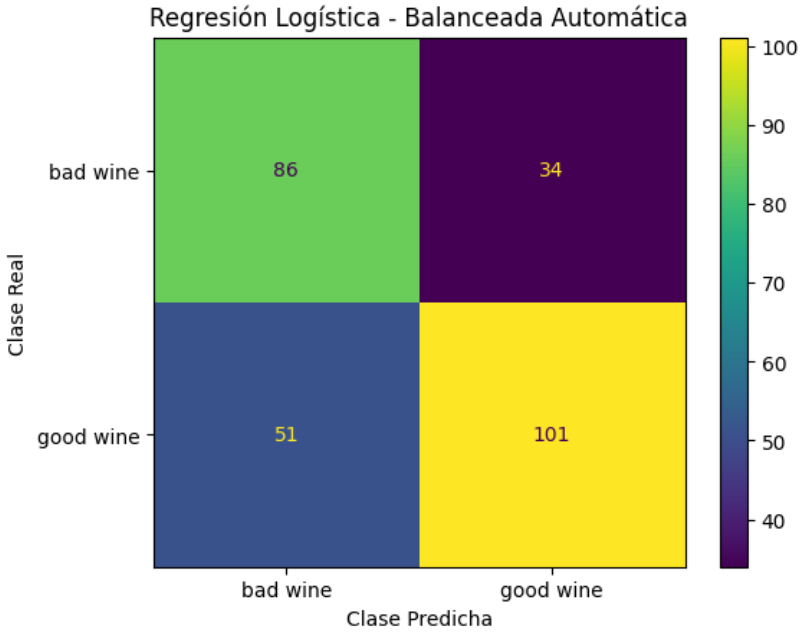
Podríamos pensar que todos los vinos predichos como buenos serán probados para ver si realmente lo son, por lo que **será prioritario predecir lo menos posible como vino malo a un vino que en realidad es bueno** ya que lo descartaríamos sin probarlo.

Con esto en mente, ahora se utiliza un modelo de **Regresión Logística con Balance manual** mediante *modelo_lr_pesos = LogisticRegression(class_weight={0:1, 1:2})*, obteniendo un F1-score de 70,4% .


Ahora, **únicamente 22 vinos son predichos como malos cuando en realidad son buenos**. Pero, 56 vinos son predichos como buenos cuando en realidad son malos. Sin embargo, conseguimos

disminuir un tipo específico de error: en este caso, aumentamos bastante el recall sobre la clase 1 (vino bueno), en decir, los encontramos a casi todos.

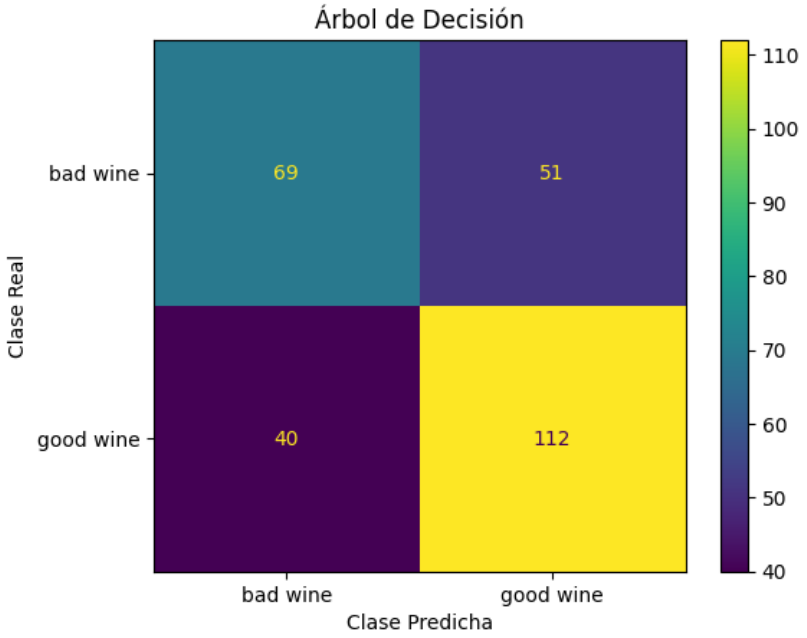
Al utilizar un modelo de **Regresión Logística con balance automático**, se obtiene un F1-score de 68,86%.




En este caso, el balance automático se parece mucho a sin balancear.

Por eso siempre es importante saber bien qué se busca, y con eso definir el enfoque a usar y cuál es el mejor modelo.

En el caso de un modelo de **Árbol de Decisión**, se obtiene un F1-score de 66,32%, por lo que parece que es un poco peor en este problema que el de Regresión Logística.




Ahora quizá tenga más sentido **analizar si hay variables que poco aportan**. Con esa información, podemos simplificar tanto el proceso de toma de datos en la vida real, como los datos a procesar, el tamaño del modelo necesario, etc.

| Index | Variables | Importancia DTC |
|---|---|---|
| **10** | alcohol | 0.278026 |
| **1** | volatile acidity | 0.139728 |
| **6** | total sulfur dioxide | 0.114684 |
| **9** | sulphates | 0.102628 |
| **3** | residual sugar | 0.066173 |
| **0** | fixed acidity | 0.061488 |
| **8** | pH | 0.054030 |
| **7** | density | 0.051516 |
| **5** | free sulfur dioxide | 0.051133 |
| **4** | chlorides | 0.046716 |
| **2** | citric acid | 0.033877 |

Observemos que **las 4 primeras variables son las que previamente consideramos más relevantes** para la variable de salida y las 7 siguientes son las que descartábamos por correlación o por ser poco relevantes para la variable de salida.

Finalmente, se aplican también los modelos de **K vecinos cercanos, Red Neuronal y SVM**, cuyos desempeños son resumidos en la siguiente tabla:

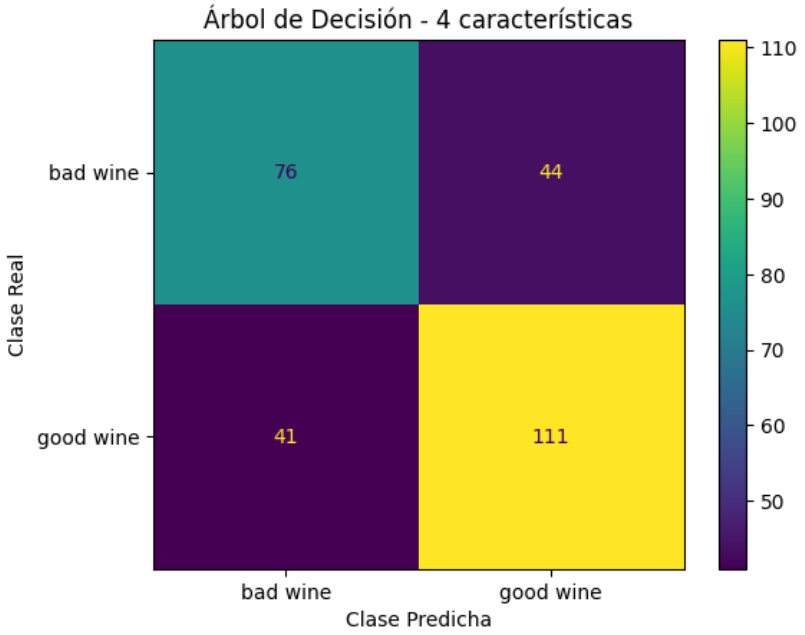
| | Accuracy | Recall | Precision | F1-Score | MCC score | Time to Train | Time to Predict | Total Time |
|---|---|---|---|---|---|---|---|---|
| LogisticRegression | 69.12% | 69.12% | 69.69% | 69.22% | 38.33% | 0.017 | 0.001 | 0.017 |
| LogisticRegression - Balanceada manual | 71.32% | 71.32% | 71.89% | 70.40% | 41.50% | 0.017 | 0.001 | 0.017 |
| LogisticRegression - Balanceada automática | 68.75% | 68.75% | 69.50% | 68.86% | 37.85% | 0.017 | 0.001 | 0.017 |
| DecisionTreeClassifier | 66.54% | 66.54% | 66.33% | 66.32% | 31.60% | 0.018 | 0.001 | 0.019 |
| KNeighborsClassifier | 68.38% | 68.38% | 68.29% | 68.32% | 35.67% | 0.003 | 0.036 | 0.039 |
| Multi-layer Perceptron classifier | 68.75% | 68.75% | 69.63% | 68.85% | 38.03% | 0.846 | 0.001 | 0.847 |
| SVM | 69.85% | 69.85% | 70.54% | 69.96% | 39.98% | 0.082 | 0.022 | 0.104 |

Podemos ver que el modelo que tiene el mejor F1-Score es la **Regresión Logística con Balanceado manual**, seguida de cerca por **SVM** pero a esta última le lleva más tiempo.

No nos olvidemos que la **Regresión Logística con Balanceado manual** además cumple con nuestro objetivo de predecir lo menos posible como vino malo a un vino que en realidad es bueno ya que lo descartaríamos sin probarlo, por lo que sin lugar a dudas este sería el modelo óptimo para resolver este problema.

## 4. PRIMEROS AJUSTES

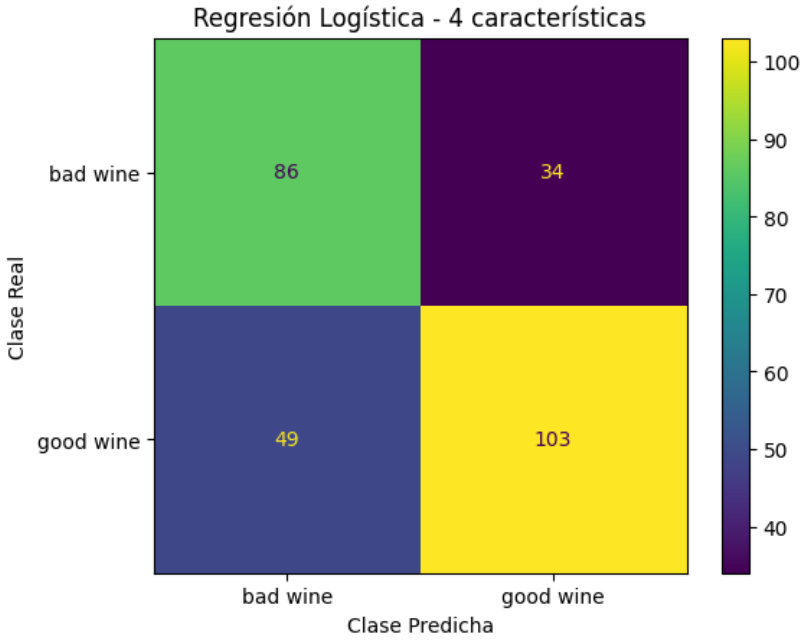
A continuación, se utiliza el modelo de **Árbol de Decisión**, pero esta vez con **sólo las 4 características más importantes** de acuerdo a lo analizado hasta ahora, que son 'alcohol', 'volatile acidity', 'total sulfur dioxide' y 'sulphates', obteniendo un F1-score de 68,7%.




El desempeño sigue siendo casi el mismo, pero **redujimos considerablemente la cantidad de datos que se usan**.

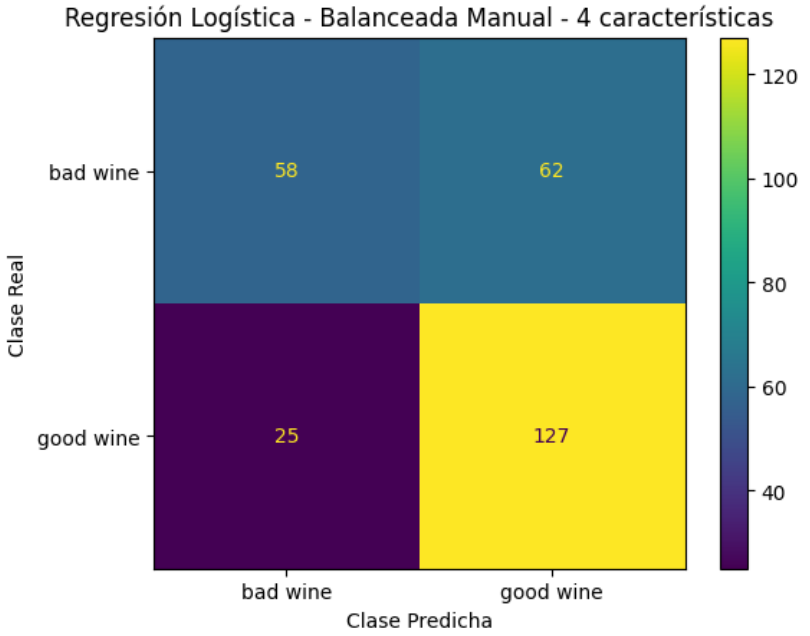
Dado que los modelos de **Regresión Logística** y **Regresión Logística con Balanceo manual** habían tenido un buen desempeño, decido ver qué pasa con ellos al sólo considerar las 4 características más importantes.

En el caso de la **Regresión Logística**, el F1-score es de 69,59%.




Vemos que nuevamente el desempeño es muy similar al de Regresión Logística con las 11 características, pero **redujimos considerablemente la cantidad de datos que se usan**.

En el caso de la **Regresión Logística con balanceo manual**, el F1-score es de 66,84%.




En este caso empeora un poco el desempeño con respecto al de Regresión Logística con Balanceado manual con las 11 características, pero sigue siendo bastante bueno.

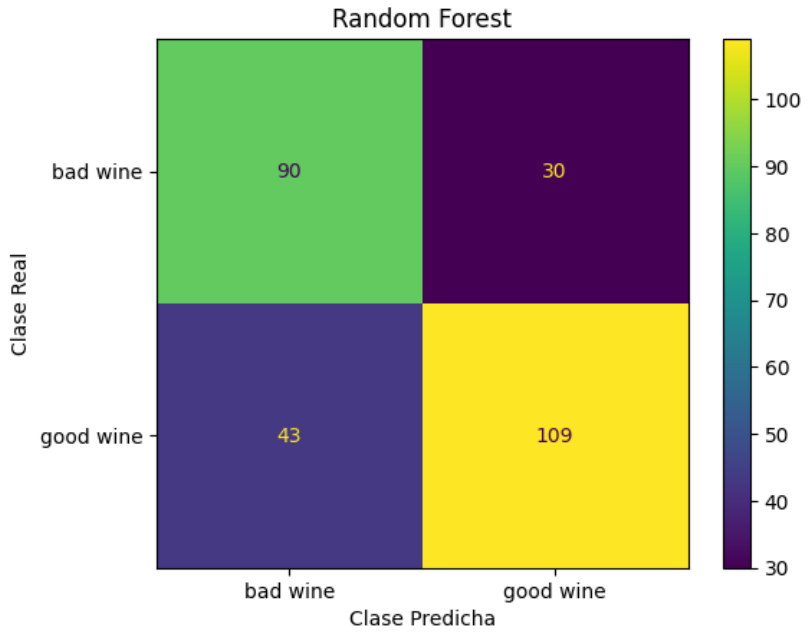
A continuación visualizamos todas las métricas:

| | Accuracy | Recall | Precision | F1-Score | MCC score | Time to Train | Time to Predict | Total Time |
|---|---|---|---|---|---|---|---|---|
| LogisticRegression | 69.12% | 69.12% | 69.69% | 69.22% | 38.33% | 0.017 | 0.001 | 0.017 |
| LogisticRegression - Balanceada manual | 71.32% | 71.32% | 71.89% | 70.40% | 41.50% | 0.017 | 0.001 | 0.017 |
| LogisticRegression - Balanceada automática | 68.75% | 68.75% | 69.50% | 68.86% | 37.85% | 0.017 | 0.001 | 0.017 |
| DecisionTreeClassifier | 66.54% | 66.54% | 66.33% | 66.32% | 31.60% | 0.018 | 0.001 | 0.019 |
| KNeighborsClassifier | 68.38% | 68.38% | 68.29% | 68.32% | 35.67% | 0.003 | 0.036 | 0.039 |
| Multi-layer Perceptron classifier | 68.75% | 68.75% | 69.63% | 68.85% | 38.03% | 0.846 | 0.001 | 0.847 |
| SVM | 69.85% | 69.85% | 70.54% | 69.96% | 39.98% | 0.082 | 0.022 | 0.104 |
| DecisionTreeClassifier - 4 características | 68.75% | 68.75% | 68.68% | 68.70% | 36.46% | 0.082 | 0.022 | 0.104 |
| LogisticRegression - 4 características | 69.49% | 69.49% | 70.12% | 69.59% | 39.16% | 0.011 | 0.001 | 0.012 |
| LogisticRegression - Balanceada manual - 4 características | 68.01% | 68.01% | 68.38% | 66.84% | 34.38% | 0.011 | 0.001 | 0.012 |

Concluimos hasta acá que si lo que se busca es precisión, el modelo ideal es de **Regresión Logística con balanceado manual**, mientras que si se busca rapidez y simplificación del modelo, se puede utilizar este modelo **pero con sólo las 4 características más importantes**.

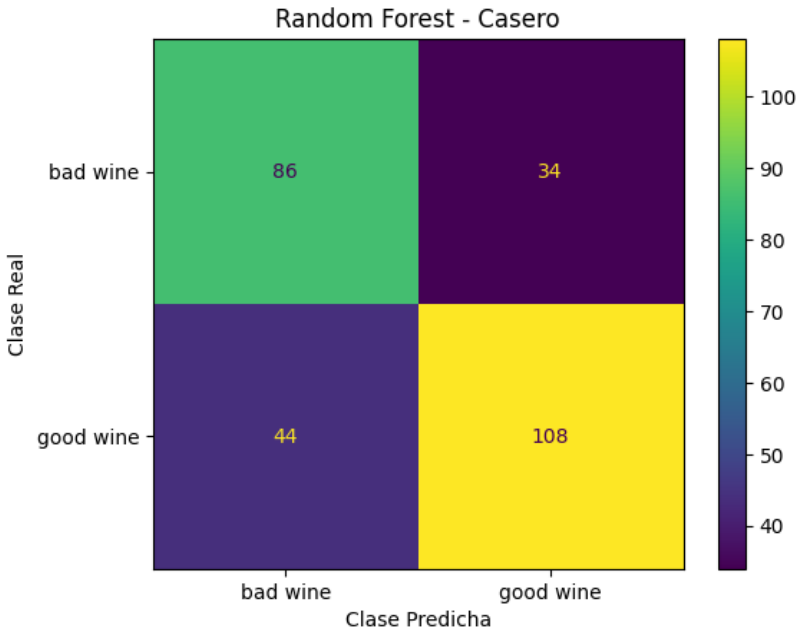
## 5. META-CLASIFICADORES

Finalmente se implementan modelos meta-clasificadores, comenzando con **Random Forest** con un F1-score de 73,25%.



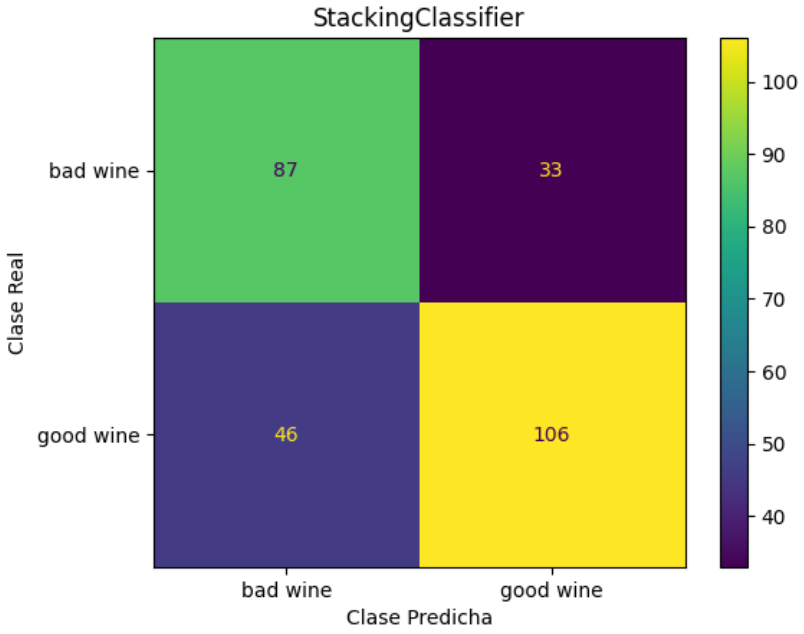

Vemos que el desempeño ha mejorado, pero hay más vinos buenos predichos como malos (43) que vinos malos predichos como buenos (30), y lo que queríamos era disminuir la primera cantidad.

Siguiendo con **Bagging Classifier**, se obtiene un F1-score de 71,41%.



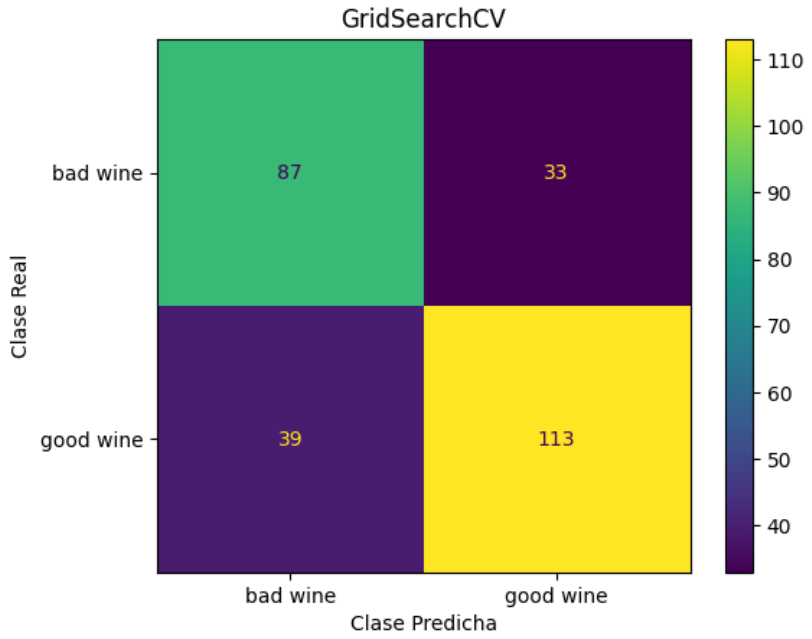

Vemos que el desempeño es similar, aunque un poco peor, con respecto al Random Forest.

A continuación, con **Stacking Classifier** se obtiene un F1-score de 71,05%.




Vemos que el desempeño es similar, aunque un poco peor, con respecto al Bagging Classifier.

Por último, con **Grid Search CV** se obtiene un F1-score de 73,59%.




Por lo tanto, de los modelos meta-clasificadores, este es el de mejor desempeño.

A continuación visualizamos todas las métricas:

| | Accuracy | Recall | Precision | F1-Score | MCC score | Time to Train | Time to Predict | Total Time |
|---|---|---|---|---|---|---|---|---|
| LogisticRegression | 69.12% | 69.12% | 69.69% | 69.22% | 38.33% | 0.023 | 0.000 | 0.024 |
| LogisticRegression - Balanceada manual | 71.32% | 71.32% | 71.89% | 70.40% | 41.50% | 0.023 | 0.000 | 0.024 |
| LogisticRegression - Balanceada automática | 68.75% | 68.75% | 69.50% | 68.86% | 37.85% | 0.023 | 0.000 | 0.024 |
| DecisionTreeClassifier | 66.54% | 66.54% | 66.33% | 66.32% | 31.60% | 0.009 | 0.000 | 0.009 |
| KNeighborsClassifier | 68.38% | 68.38% | 68.29% | 68.32% | 35.67% | 0.003 | 0.034 | 0.037 |
| Multi-layer Perceptron classifier | 70.22% | 70.22% | 70.74% | 70.32% | 40.47% | 0.821 | 0.001 | 0.822 |
| SVM | 69.85% | 69.85% | 70.54% | 69.96% | 39.98% | 0.077 | 0.023 | 0.099 |
| DecisionTreeClassifier - 4 características | 68.75% | 68.75% | 68.68% | 68.70% | 36.46% | 0.077 | 0.023 | 0.099 |
| LogisticRegression - 4 características | 69.49% | 69.49% | 70.12% | 69.59% | 39.16% | 0.008 | 0.001 | 0.009 |
| LogisticRegression - Balanceada manual - 4 características | 68.01% | 68.01% | 68.38% | 66.84% | 34.38% | 0.008 | 0.001 | 0.009 |
| Random Forest | 73.16% | 73.16% | 73.68% | 73.25% | 46.40% | 0.008 | 0.001 | 0.009 |
| Random Forest - Casero | 71.32% | 71.32% | 71.69% | 71.41% | 42.46% | 0.008 | 0.001 | 0.009 |
| StackingClassifier | 70.96% | 70.96% | 71.47% | 71.05% | 41.95% | 0.008 | 0.001 | 0.009 |
| GridSearchCV | 73.53% | 73.53% | 73.71% | 73.59% | 46.64% | 0.008 | 0.001 | 0.009 |

Concluimos entonces que:

- Si lo que se busca es precisión, el modelo ideal es de **Grid Search CV**, aunque hay que considerar que le llevó aproximadamente 10 minutos encontrar los mejores parámetros.
- En segundo lugar, sigue el **Random Forest** que con una precisión similar, es mucho más rápido.
- Finalmente, si lo que se busca es cumplir con el objetivo de predecir lo menos posible como vino malo a un vino que en realidad es bueno, entonces lo ideal es utilizar **Regresión Logística con balanceado manual.**